DETAILED ACTION

1. This action is responsive to claims filed 03/21/2008 and Applicant's request for reconsideration of application 10/774038 filed 03/21/2008.

The amendment contains amended claims 2, 3, 12-18, 20, 22, 23, 25, 27, and 28.

Claims 1, 4-11, 19, 21, 24, and 26 have been canceled.

Information Disclosure Statement

2. The information disclosure statement filed 01/29/2008 has been received, considered as indicated, and placed on record in the file.

Claim Objections

3. The claims are objected to because of the following minor informalities:

a. Claim 28 should be corrected such that it ends with a "." and not a ",".

Appropriate correction is required.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless – (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application

filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

5. Claims 2, 3, 12-14, 20, 22, 23, 25, 27, and 28 are rejected under 35 U.S.C. 102(e) as being anticipated by Rudd et al. (PGPub No. 20030233394).

As per claim 13, Rudd et al. teaches a **thread selection unit** (thread control logic [Figure 2, element 270], whose details are shown in [Figure 3]) **for a multithreaded processor** ([Abstract, line 2])**, the thread selection unit comprising: a priority thread selector** (thread prioritizer [Figure 3, element 310] and switch enabler [Figure 3, element 320]) **configured to generate a priority thread value** (absolute thread priority, [Figure 3, element 354]) **associated with a priority thread selected from a plurality of active threads in the multithreaded microprocessor** ([Abstract, line 2])**; the priority thread selector comprising: a plurality of maxtime registers** (threshold values [Figure 5, element 522], where storage of threshold values in registers for use in counter comparisons is old and well known in the art)**, wherein each active thread has an associated maxtime register** (forward progress counters [Figure 3, elements 322 and 324], which are associated with each active thread [paragraph 28, lines 1-2]. The forward progress counters count events which may include weighted processor cycles [paragraph 28, lines12-29]. Thresholds are used to determine when a forward progress counter has reached a limit where the priority thread may be switched. Each threshold value can be associated with a forward progress counters [paragraph 35, lines 12-21] and [paragraph 36, lines 5-9])**; an internal thread counter** (thread prioritizer [Figure 3, element 310]) **configured to provide an internal thread value to the maxtime registers** (the thread prioritizer has corresponding forward progress counters [paragraph 28, lines 1-2]. Therefore, when a thread is the active thread the corresponding forward progress counter is active) **and receive a maxtime value** (threshold [Figure 5, element 522]) **corresponding to the internal thread value** (thresholds can be associated with individual forward progress counters [paragraph 35, lines 12-21] and [paragraph 36, lines 5-9] which have corresponding threads [paragraph 28, lines 1-2])**; a counter** (forward progress counters [Figure 3, elements 322 and 324])**; and a comparator coupled to the counter, the internal thread counter and the plurality of maxtime registers, wherein the comparator is configured to compare a count value of the counter with the maxtime value** (one skilled in the art would recognize that a comparator must be used when Rudd et al. performs a comparison [paragraph 36, lines 4-12] of the forward progress counters, which are associated with each

thread [paragraph 28, lines 1-2], to its associated threshold value [paragraph 35, lines 12-13]); **and an execution thread selector** (next thread selection [Figure 3, element 330]) **coupled to receive the priority thread value and to generate an execution thread value associated with an execution thread** (next threads [Figure 3, element 360]), **wherein the internal thread counter is incremented** (a new thread can be switched in [paragraph 35 lines 7-10]) **and the counter is reset** (initialized to zero, [Figure 8, element 810] and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized) **when the maxtime value equals zero** ([Figure 5], where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 - paragraph 36 line 22]. If the forward progress counter is zero, the thread can be switched immediately.).

As per claim 2, the rejection of claim 13 has been addressed.
Rudd et al. teaches a thread selection unit **wherein the execution thread selector is configured to select the priority thread as the execution thread when the priority thread is unblocked** ([paragraph 29, lines 13-17], where "the current thread waiting for memory or I/O access" is an example of a blocked thread).

As per claim 3, the rejection of claim 13 has been addressed.
Rudd et al. teaches a thread selection unit **wherein the priority thread selector selects the priority thread without regards to the actions of the execution thread selector** ([Figure 3], where next threads output from next thread selector is not an input to either the thread prioritizer or switch enabler).

As per claim 12, the rejection of claim 13 has been addressed.
Rudd et al. teaches a thread selection unit **wherein the internal thread counter is incremented** (a new thread can be switched in [paragraph 35 lines 7-10]) **and the counter is reset** (initialized to zero [Figure 8, element 810] and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized) **when the count value equals the maxtime value** ([Figure 5], where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 - paragraph 36 line 22]).

As per claim 14, the rejection of claim 12 has been addressed.
Rudd et al. teaches a thread selection unit **wherein the priority thread selector further comprises a priority thread register** (thread prioritizer [Figure 3, element 310], which contains a register of prioritized threads) **coupled to the comparator and configured to receive the internal thread value** (the thread

prioritizer performs the functions of both the internal thread counter and priority thread register) **and then provide the priority thread value** (absolute thread priority [Figure 3, element 354])**, wherein the priority thread registers stores the internal thread value when the maxtime value is not equal to zero** ([Figure 5], where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 – paragraph 36 line 22]. If the forward progress counter is zero, the thread can be switched immediately.)**.**

As per claim 23, Rudd et al. teaches a method ([claim 1])
All of the limits of Claim 23 have been previously addressed in Claims 2, 12, 13, and 14, and is therefore rejected using the same prior art and rationale.

As per claim 20, the rejection of claim 19 has been addressed.
Rudd et al. further teaches **a method further comprising selecting a non-priority thread as the execution thread when the priority thread is blocked** ([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)**.**

As per claim 22, the rejection of claim 21 has been addressed.
Rudd et al. further teaches **a method wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: of incrementing a priority thread counter** (a new thread can be switched in [paragraph 35 lines 7-10]) **when a count value equals the maxtime value** ([Figure 5], when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 - paragraph 36 line 22]) **corresponding to the priority thread; and resetting a counter when the count value equals the maxtime value corresponding to the priority thread** (initialized to zero [Figure 8, element 810] and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized)**.**

All of the remaining limits of Claim 22 have been previously addressed in Claim 13 and is therefore rejected using the same prior art and rationale.

As per claim 28, All of the limits of Claim 28 have been previously addressed in Claims 2, 12, 13, 14, and 23, and is therefore rejected using the same prior art and rationale.

As per claim 25, the rejection of claim 28 has been addressed.
All of the limits of Claim 25 have been previously addressed in Claim 20, and is therefore rejected using the same prior art and rationale.

As per claim 27, the rejection of claim 28 has been addressed.
All of the limits of Claim 27 have been previously addressed in Claim 22, and is therefore rejected using the same prior art and rationale.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 15-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rudd et al. (PGPub No. 20030233394) in view of Borkenhagen et al. (U.S. Patent No. 6,567,839).

As per claim 15, Rudd et al. teaches a thread selection unit **wherein the execution thread value** (next thread [Figure 3, element 360]) **is generated from the execution thread selector** (next thread selection, [Figure 3, element 330]).

Rudd et al. does not teaches **unit wherein the execution thread selector comprises: a thread block checker configured to provide a plurality of block values, wherein each active thread has a corresponding block value; an execution thread register configured to provide the execution thread value**.

Borkenhagen et al. teaches a thread selection **unit wherein the execution thread selector** (thread switch logic hardware [Figure 4A and B, element 400]) **comprises: a thread block checker** (thread switch control register [Figure 4A and 4B, element 410]) **configured to provide a plurality of block values** ([column 12, line 64 - column 13 line 3]), **wherein each active thread has a corresponding block value** (column 13 lines 2-3]); **an execution thread register** (storage control unit [Figure 4B, element 200], where the execution thread register are part of the instruction pipeline for execution of instructions) **configured to provide the execution thread value** (input "A" in Figures 4A and output "A" in Figure 4B);

It would have been obvious to one skilled in the art at the time of the invention to have used the thread switch unit implemented in Borkenhagen et al. as the next thread selection unit of Rudd et al. Further, it would have been obvious to one skilled in that art that ("The next thread selector "[Figure 3, element 330] could be combined with the thread switch logic of Borkenhagen et al.[Figures 4A and 4B] to provide a simple comparison using a comparator, which would be known to someone skilled in the art, of the priority thread value (Rudd et al., absolute thread value [Figure 3, element 354]) and the execution thread value (Borkenhagen et al. input "A" in Figures 4A and output "A" in Figure 4B). The purpose of using Borkenhagen et al. is to provide an improved thread selection system which can reduce delays due to memory latency in a multilevel cache system utilized in conjunction with a multithread data processing system.

All of the remaining limits of Claim 15 have been previously addressed in Claim 13, and is therefore rejected using the same prior art and rationale.

As per claim 16, the rejection of claim 15 has been addressed.
Rudd et al. teaches **a thread selection unit to receive the priority thread values** (absolute thread (element 354, Figure 3)) **that generates a next execution thread value for the execution thread register** (next threads (element 360, Figure 3).

Rudd et al. does not teach the thread selection unit **wherein the execution thread selector further comprises a controller coupled to receive the block values, the comparison result, and the execution thread value**.

Borkenhagen et al teaches **the thread selection unit wherein the execution thread selector further comprises a controller** (thread switch controller [Figure 4A,element 450]) **coupled to receive the block values** (thread switch control register is coupled to the thread switch controller in [Figure 4A]) coupled to receive the block values, the priority thread value, the comparison result, and the execution thread value and configured to generate a next execution thread value for the execution thread register. (thread switch controller [Figure 4A,element 450] is shown receiving all inputs required for determining the active thread.)

It would have been obvious to one skilled in the art at the time of the invention to have used the thread switch controller in Borkenhagen et al. as the next thread selection unit of Rudd et al. The purpose would have been able to use a controller to make logical decisions based the all available thread status (block information, the priority thread value, the current thread being executed, and all active threads). Note that next thread selector (element 330) of Figure 3 in Rudd et al. has all information available to it in the execution information (element 352) and configuration (element 350) for implementing the logic of Borkenhagen et al.

The purpose of using Borkenhagen et al. is to provide an improved thread selection system which can reduce delays due to memory latency in a multilevel cache system utilized in conjunction with a multithread data processing system.

As per claim 17, the rejection of claim 16 has been addressed. Rudd et al. further teaches a thread selection unit **wherein the controller generates the next execution thread value to be equal to the priority thread value when the priority thread is not blocked.** ([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

As per claim 18, the rejection of claim 17 has been addressed. Rudd et al. further teaches a thread selection unit **wherein the controller generates the next execution thread value to not be equal to the priority thread value when the priority thread is blocked** ([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread).

Response to Arguments

8. Applicant's arguments, with regards to claims 2, 3, 12-18, 20, 22, 23, 25, 27, and 28, filed 03/21/2008 have been fully considered but they are not persuasive.

9. On page 8 of the Applicant's Response, the applicants argue that Rudd et al. does not disclose "setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero" as claimed in amended claims 13, 23, and 28. The applicants further state that Rudd et al. does not disclose checking whether the threshold or limit of a thread's forward progress counter, which is equivalent to the maxtime register value, equals zero prior to designating the thread as the currently executing (i.e., priority) thread and assert that claims 13, 23, and 28 recite a limitation that would not permit such a thread to become the priority thread.

10. The Examiner disagrees with the Applicant's arguments concerning Rudd et al. The claim limit cited indicates what the thread selector does when the "internal thread value is not equal to zero". There is no limit that prevents "setting a priority thread value equal to the internal thread value" when the internal thread value is equal to zero. Further, the applicant has not claimed "checking whether the maxtime register value equals zero prior to designating the thread as the currently executing (priority) thread". Therefore, this argument is moot.

11. On page 9 of the Applicant's Response, the applicants further argue that Borkenhagen et al. does not disclose a "block value corresponding to each active thread". The applicants further asserts that "the thread switch control register" (taught in Borkenhagen et al.) "stores a plurality of block values for each thread such that if one of those block values is determined to correspond to a given thread, the thread would be considered blocked." and that "the thread switch control register does not actually provide the corresponding block value as required by amended claim 15".

12. The Examiner respectfully disagrees with Applicant's argument. Borkenhagen et al. discloses that a single bit is associated with each thread is used to enable and disable a thread switch ([column 16, lines 55-62]). Further, Borkenhagen et al. discloses that separate thread switch control registers for each thread could be used to add flexibility.

13. The use of Vaitzblit et al. as grounds of rejection is retracted.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Vaitzblit et al. (U.S. Patent No. 5,528,513) – teaches the use of an execution thread selector.
- Sager (PGPub No. 20030154235) – teaches a method and apparatus for controlling the processing priority between multiple threads in a multithreaded processor.

15. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Gregory Pollock whose telephone number is 571 270-1465. The examiner can normally be reached on 7:30 AM - 4 PM, Mon-Fri Eastern Time.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jay Kramer can be reached on 571 272-6783. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service

Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

GAP

6/9/2008

/Gregory Pollock/

Examiner, Art Unit 3693